A TSK-Type-Based Self-Evolving Compensatory Interval Type-2 Fuzzy Neural Network (TSCIT2FNN) and Its Applications

Yang-Yin Lin, Jyh-Yeong Chang, *Member, IEEE*, and Chin-Teng Lin, *Fellow, IEEE*

***Abstract*—In this paper, a Takagi–Sugeno–Kang (TSK)-type-based self-evolving compensatory interval type-2 fuzzy neural network (FNN) (TSCIT2FNN) is proposed for system modeling and noise cancellation problems. A TSCIT2FNN uses type-2 fuzzy sets in an FNN in order to handle the uncertainties associated with information or data in the knowledge base. The antecedent part of each compensatory fuzzy rule is an interval type-2 fuzzy set in the TSCIT2FNN, where compensatory-based fuzzy reasoning uses adaptive fuzzy operation of a neural fuzzy system to make the fuzzy logic system effective and adaptive, and the consequent part is of the TSK type. The TSK-type consequent part is a linear combination of exogenous input variables. Initially, the rule base in the TSCIT2FNN is empty. All rules are derived according to online type-2 fuzzy clustering. For parameter learning, the consequent part parameters are tuned by a variable-expansive Kalman filter algorithm to the reinforce parameter learning ability. The antecedent type-2 fuzzy sets and compensatory weights are learned by a gradient descent algorithm to improve the learning performance. The performance of TSCIT2FNN for the identification is validated and compared with several type-1 and type-2 FNNs. Simulation results show that our approach produces smaller root-mean-square errors and converges more quickly.**

***Index Terms*—Compensatory operation, fuzzy identification, online fuzzy clustering, type-2 fuzzy systems.**

I. Introduction

RECENTLY, fuzzy neural networks (FNNs) have become popular in applications in control, identification, prediction, pattern recognition, and bioengineering. FNNs inherit their learning ability from neural networks and their inference technology from fuzzy systems and are used for solving the aforementioned characteristic behaviors [1]–[12], such as in the control of robot manipulators [4], temperature control [5], pattern classification [6], ventricular premature contraction (VPC) detection [7], energy conversion [8], and hardware implementation [9]. FNNs are an effective tool for dealing with complex nonlinear processes. Some well-known feedforward FNNs include an adaptive neuro-fuzzy inference system [2], an online self-constructing neural fuzzy inference network (SONFIN) [3], and fuzzy wavelet neural networks (FWNNs) [10]–[12].

Moreover, the aforementioned FNNs use only a static fuzzy inference mechanism, such as a Min or Max operator, to achieve fuzzy reasoning. There are two disadvantages to using fixed fuzzy operators. The first is that there are not enough free variables to adjust the fuzzy operator, and the second is that fixed fuzzy operators cannot optimize the fuzzy logic reasoning. In [13], Zimmermann and Zysno first defined the essence of a compensatory operator in order to the overcome aforementioned disadvantages. The enhanced compensatory operation was presented according to whether a pessimistic or optimistic operation was to be taken [14]. Many studies [15]–[19] have dealt with the optimization of the fuzzy membership function (MF), as well as defuzzification via parameter learning algorithms. In [20]–[24], compensatory attempts were widely used in a variety of FNNs. As there are fewer type-2 fuzzy logic systems (FLSs) using a compensatory fuzzy inference mechanism, we will extend the compensatory mechanism to type-2 FLSs in this study.

The type-2 fuzzy set is an extension of the type-1 fuzzy set, as first introduced by Zadeh [1]. Theoretical developments in type-2 sets have been widely reported in the literatures [25]–[29]. Indeed, studies on type-2 FLSs [30]–[33] have drawn great attention because type-2 fuzzy sets seem to be a more effective approach for handling uncertainties in the rule base associated with information and data, with successful applications in a variety of areas, such as control [34]–[36], medical applications [37], and noise tolerance analysis [38]. However, generalized type-2 FLSs are more complex than type-1 FLSs. The theory of interval type-2 fuzzy sets has been proposed [26] in order to improve the computational complexity and load. Many researchers have considered using interval type-2 fuzzy systems. Therefore, the design of an interval type-2 FNN has drawn considerable efforts. In [39]–[42] and [65], the authors proposed the parameter learning of an interval type-2 fuzzy system using Gaussian primary MFs with uncertain means. In [39], the optimal training of an interval type-2 FNN using the gradient descent algorithm is discussed. In [43], a hybrid learning algorithm for interval type-2 FNNs is presented. Recently, some interval type-2 FNNs have been proposed for the automatic design of type-2 FLSs [44]–[51]. In [44], a

Manuscript received March 23, 2012; revised July 10, 2012, September 15, 2012, and November 27, 2012; accepted February 14, 2013. Date of publication February 22, 2013; date of current version July 18, 2013. This work was supported in part by the Aiming for the Top University Plan of National Chiao Tung University, and in part by the Ministry of Education, Taiwan, under Contract 101W963. Further support came from the University System of Taiwan-University of California, San Diego, International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council International Research-Intensive Centers of Excellence Program under Grant NSC-100-2911-I-009-101.

Y.-Y. Lin and J.-Y. Chang are with the Institute of Electrical Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: oliver.yylin@gmail.com).

C.-T. Lin is with the Institute of Electrical Control Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan, and also with the Brain Research Center, National Chiao Tung University, Hsinchu 300, Taiwan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2013.2248332

self-evolving interval type-2 FNN (SEIT2FNN) model uses interval type-2 sets with uncertain means in the antecedent parts and interval type-1 Takagi–Sugeno–Kang (TSK) fuzzy sets in the consequent parts. In a further study described in [46], the type-2 FNN uses two forms of interval type-2 fuzzy sets, one with uncertain means but fixed standard deviations (STDs) and the other with uncertain STDs but a fixed mean, and compares their performances. Some type-2 FNNs [44], [46], [51] are trained using a Kalman filter algorithm to improve the learning performance. In [48], the author proposed a novel type-2 TSK-based fuzzy neural structure (FNS) for the identification and control of dynamic plants. The design of two-axis motion control using an interval type-2 fuzzy set was proposed in [49], and the structure of a discrete interval type-2 fuzzy system by fuzzy c-means (FCM) clustering has been proposed in [50].

In industry, most plants are susceptible to uncertainties in internal and external disturbances and focus on time-varying technique. Therefore, the purpose of this study is to develop an interval type-2 FNN, i.e., a TSK-type-based self-evolving compensatory interval type-2 FNN (TSCIT2FNN), for system modeling, problem for forecasting, and noise cancellation. The self-evolving property means that the TSCIT2FNN can automatically evolve the required network structure (rule number and initial fuzzy set shape) and parameters according to pieces of training data. In addition, the self-evolving property enables the efficient handling of the time-varying characteristic problem. In this paper, the major contributions of the TSCIT2FNN are twofold. The first is the proposal of a compensatory operator in the type-2 inference mechanism. To use the compensatory operator, an inference mechanism in the TSCIT2FNN is more flexible for optimizing the fuzzy reasoning via a gradient descent algorithm. For our second contribution, this paper proposes a variable-expansive Kalman filter algorithm for the learning of consequent weights in order to reinforce the learning accuracy.

The rest of this paper is organized as follows. Section II introduces the TSCIT2FNN structure, which combines a compensatory operation in the inference mechanism and a crisp TSK function in the consequent part. Section III presents the structure learning of the TSCIT2FNN and its parameter update rules. Section IV describes five simulation examples, including three types of system identification, adaptive noise cancellation (ANC), and time-series prediction. Section V contains our conclusions.

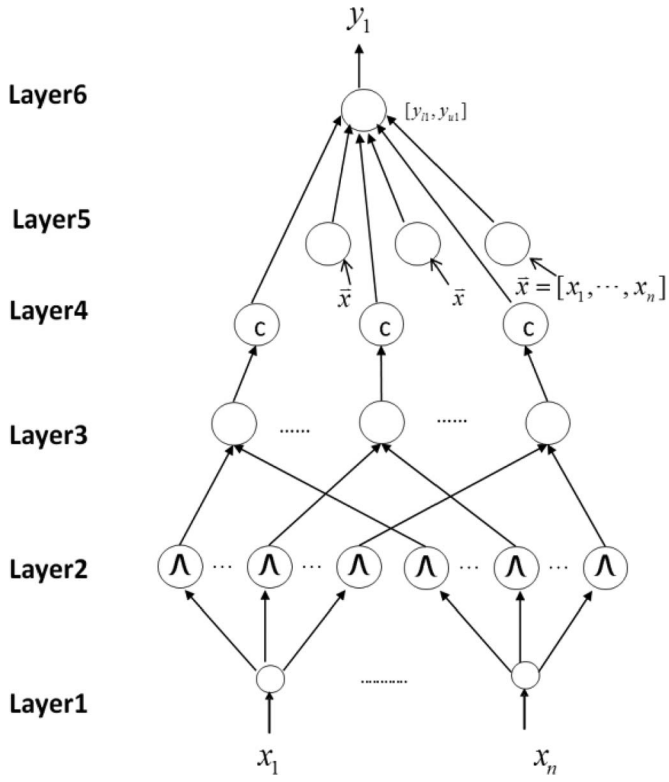

Fig. 1. Proposed six-layer TSCIT2FNN structure, where each compensatory rule in layer 4 formulates a compromise between the optimistic and the pessimistic operator and each node in layer 5 performs a linear combination of current input states.

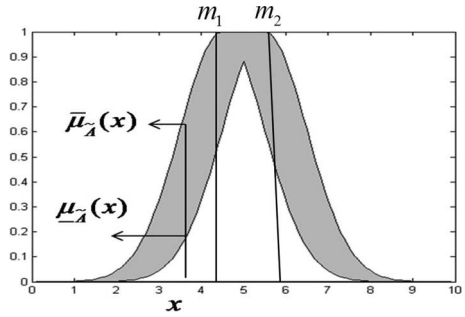

Fig. 2. Interval type-2 fuzzy set with uncertain mean.

II. TSCIT2FNN Structure

This section introduces the structure of a multiple-input single-output TSCIT2FNN. Fig. 1 shows the proposed six-layered TSCIT2FNN structure. The premise parts of the TSCIT2FNN utilize the interval type-2 fuzzy sets having uncertain means and fixed STDs, and the consequent part of each compensatory fuzzy rule is of the TSK type and executes a crisp linear model. The proposed TSCIT2FNN realizes the fuzzy if-then rules in the following form:

$$\text{Rule } i: \text{IF} \left(x_1 \; is \; \tilde{A}_1^i \; and \; \ldots \; and \; x_n \; is \; \tilde{A}_n^i\right)^{1-\gamma^i+\frac{\gamma^i}{n}} \quad \text{THEN} \quad y \; is \quad a_0^i + \sum_{j=1}^{n} a_j^i x_j \qquad i = 1, \ldots, M \quad (1)$$

where x_j, $j = 1, \ldots, n$, denotes input variables and y is the output variable. $\tilde{A}_n^i$ denotes type-2 MFs used by the Gaussian MF. $\gamma^i \in [0, 1]$ is a compensatory degree, which can not only adaptively adjust fuzzy MFs but also dynamically optimize the adaptive fuzzy reasoning. a_0^i and a_j^i for the ith rule of the jth input are TSK-type consequent part parameters, and M is the total number of rules. The detailed mathematical functions of each layer are introduced hereinafter.

Layer 1 (input layer): Each node in this layer is a crisp input variable. Only input variables are fed into this layer. Note that there are no weights to be adjusted in this layer.

Layer 2 (MF layer): Each node in this layer utilizes an interval type-2 MF to perform the fuzzification operation. As stated earlier, the interval type-2 fuzzy set $\tilde{A}_j^i$ uses a Gaussian

primary MF having a fixed STD σ and an uncertain mean that takes values in $[m_1, m_2]$ (see Fig. 2), i.e.,

$$\mu_{\tilde{A}_j^i} = \exp\left\{-\frac{1}{2}\left(\frac{x_j - m_j^i}{\sigma_j^i}\right)^2\right\} \equiv N\left(m_j^i, \sigma_j^i; x_j\right), \quad m_j^i \in \left[m_{j1}^i, m_{j2}^i\right]. \tag{2}$$

The footprint of uncertainty (FoU) of this MF can be represented as a bounded interval in terms of an upper MF, $\bar{\mu}_j^i$, and a lower MF, $\underline{\mu}_j^i$, where

$$\overline{\mu}_j^i(x_j) = \begin{cases} N\left(m_{j1}^i, \sigma_j^i; x_j\right), & x_j < m_{j1}^i \\ 1, & m_{j1}^i \le x_j m_{j2}^i \\ N\left(m_{j2}^i, \sigma_j^i; x_j\right), & x_j > m_{j2}^i \end{cases} \tag{3}$$

$$\underline{\mu}_j^i(x_j) = \begin{cases} N\left(m_{j2}^i, \sigma_j^i; x_j\right), & x_j \le \frac{m_{j1}^i + m_{j2}^i}{2} \\ N\left(m_{j1}^i, \sigma_j^i; x_j\right), & x_j > \frac{m_{j1}^i + m_{j2}^i}{2}. \end{cases} \tag{4}$$

That is, the output of each node can be represented as an interval $[\underline{\mu}_j^i, \bar{\mu}_j^i]$.

Layer 3 (firing layer): Each node in this layer represents its fuzzy rule and functions. To acquire firing strength F^i, each node performs a fuzzy meet operation with inputs from layer 2 using an algebraic product operation. The firing strength is an interval type-1 fuzzy set and is computed as follows [27]:

$$F^i = [\underline{f}^i, \overline{f}^i], \qquad i = 1, \ldots, M \tag{5}$$

$$\overline{f}^i = \prod_{j=1}^{n} \overline{\mu}_j^i, \qquad \underline{f}^i = \prod_{j=1}^{n} \underline{\mu}_j^i. \tag{6}$$

Layer 4 (compensatory firing layer): Each node in this layer has its corresponding firing strength, which may be reinforced or weakened. The t-norm fuzzy operation is a pessimistic operation as the min or product operation. The t-conorm fuzzy operation is an optimistic operation as the max operation. Therefore, the use of a compensatory operation can map the pessimistic input x_1 and the optimistic input x_2 to give a relative compromise for superior and inferior situations. For example, $c(x_1, x_2) = x_1^{1-\gamma} x_2^{\gamma}$, where γ is a compensatory degree. Here, the pessimistic value x_1 denotes f, and the optimistic value x_2 denotes $f^{1/n}$. As a result, each node in this layer is performed by a pessimistic and optimistic operation, which is called the compensatory operator. Here, the output function of the compensatory operation is denoted as

$$\psi^i(P, O) = \left[\underline{\psi}^i\left(\underline{p}(t), \underline{o}(t)\right), \overline{\psi}^i\left(\overline{p}(t), \overline{o}(t)\right)\right], \quad i = 1, \ldots, M \tag{7}$$

where

$$\underline{\psi}^i\left(\underline{p}(t), \underline{o}(t)\right) = \underline{p}^i(t)^{1-\gamma^i(t)} \times \underline{o}^i(t)^{\gamma^i(t)} = \underline{f}^i(t)^{1-\gamma^i(t)+\frac{\gamma^i(t)}{n}} \tag{8}$$

$$\overline{\psi}^i\left(\overline{p}(t), \overline{o}(t)\right) = \overline{p}^i(t)^{1-\gamma^i(t)} \times \overline{o}^i(t)^{\gamma^i(t)} = \overline{f}^i(t)^{1-\gamma^i(t)+\frac{\gamma^i(t)}{n}}. \tag{9}$$

The pessimistic operation of $P = [\underline{p}(t), \overline{p}(t)]$ at time t is expressed as

$$\underline{p}^i(t) = \underline{f}^i(t), \quad \overline{p}^i(t) = \overline{f}^i(t) \tag{10}$$

and the optimistic operation of $O = [\underline{o}(t), \overline{o}(t)]$ at time t is expressed as

$$\underline{o}^i(t) = \left(\underline{f}^i(t)\right)^{\frac{1}{n}}, \quad \overline{o}^i(t) = \left(\overline{f}^i(t)\right)^{\frac{1}{n}}. \tag{11}$$

Here, $\gamma^i(t) = c_i^2(t)/c_i^2(t) + d_i^2(t) \in [0, 1]$ is called the compensatory degree, and c_i, $d_i \in [-1, 1]$. The update parameters c_i and d_i are used to enhance the adaptive compensatory operator.

Layer 5 (consequent layer): Each node in this layer is called a consequent node and describes a linear model with exogenous inputs. Each compensatory rule node in layer 4 has a corresponding consequent node in layer 5. The output of a consequent node is a linear combination of external input states denoted by

$$y_{\text{TSK}}^i = a_0^i + \sum_{j=1}^{n} a_j^i x_j = a_0^i x_0 + a_n^i x_1 + \cdots + a_n^i x_n \tag{12}$$

where $x_0 \equiv 1$, $x_1, \ldots, x_n$ are input variables, and the consequent part parameters are described by (1).

Layer 6 (output layer): Nodes in this layer correspond to one output linguistic variable. The output function combines the output of layers 3 and 4, and the design factor q enables us to share the upper and the lower values in the final output. Thus, there is no need to apply the Karnik–Mendel (KM) iterative procedure [42] to find the end points. The defuzzified output is given by

$$y' = q\underline{y}' + (1-q)\overline{y}' = \frac{q\sum_{i=1}^{M} \underline{\psi}^i y_{\text{TSK}}^i}{\sum_{i=1}^{M} \underline{\psi}^i} + \frac{(1-q)\sum_{i=1}^{M} \overline{\psi}^i y_{\text{TSK}}^i}{\sum_{i=1}^{M} \overline{\psi}^i} \tag{13}$$

where $\underline{\psi}^i$ and $\overline{\psi}^i$ are computed by (8) and (9), and the design factor q can adjust the proportion of the upper and the lower, depending on the certainty of the system.

III. TSCIT2FNN Learning

This section explicitly introduces the structure and parameter learning algorithm. Initially, the rule base is empty in the TSCIT2FNN. All of the rules and free parameters are generated and derived based on simultaneous structure and parameter learning.

A. Structure Learning

The online structure learning is generated by a type-2 fuzzy clustering method. A previous study [3] proposed the criterion of type-1 rule generation according to the rule firing strength. This idea protocol is extended to type-2 fuzzy clustering as

a criterion of rule generation in the TSCIT2FNN. The firing strength F^i in (5) is used to determine whether a new rule should be generated. The type-2 rule firing strength is an interval, with the center of the firing interval (the criterion for rule generation) denoted as

$$f_c = \frac{1}{2}\left(\overline{f}^i + \underline{f}^i\right). \tag{14}$$

Next, we discuss the structure-learning algorithm of the TSCIT2FNN. On the arrival of the first incoming data x, the new fuzzy rule is generated immediately. The first uncertain means and fixed width of MFs associated with this rule are set as

$$\begin{aligned}\left[m_{j1}^1, m_{j2}^1\right] &= [x_j - \Delta x, x_j + \Delta x] \\ \sigma &= \sigma_{\text{fixed}}, \qquad j = 1, \ldots, n\end{aligned} \tag{15}$$

where σ_{fixed} is a predefined value (we use $\sigma_{\text{fixed}} = 0.4$ in this paper) that determines the width of the memberships associated with a new rule. Subsequently, for each new incoming data item $x(t)$, we find

$$I = \arg \max_{1 \le i \le M(t)} f_c^i(t) \tag{16}$$

where $M(t)$ is the number of existing rules at time t. If $f_c^I(t) \le f_{\text{th}}$ (f_{th} is a prespecified threshold), then a new fuzzy rule is generated. The idea is that the present data point is distinct from any of the existing rules, and hence, a new rule is generated. We use the same procedure to assign the uncertain means and width of the generated MF of a new rule. The uncertain means of corresponding type-2 fuzzy sets are defined as

$$\left[m_{j1}^{M(t)+1}, m_{j2}^{M(t)+1}\right] = [x_j(t) - \Delta x,\ x_j(t) + \Delta x]. \tag{17}$$

The width of the new rule is defined as follows:

$$\sigma_j^{M(t)+1} = \beta \cdot \left| x_j - \left(\frac{m_{j1}^I + m_{j2}^I}{2} \right) \right|. \tag{18}$$

Equations (15) and (17) indicate that Δx means the width of the uncertain region where the proper uncertainty region can adapt to the network input range. If the uncertainty associated with the mean is too small, then the type-2 fuzzy sets become similar to type-1 fuzzy sets. On the other hand, if the width of the uncertain region is too large, then the uncertain mean covers most of the input domain. Equation (18) uses the Euclidean distance between the current input data x and the center of the best matching rule for this data point multiplied by an overlapping parameter β to compute the width of the new type-2 fuzzy sets. In this paper, β is set to 0.5 so that the width of a new type-2 fuzzy set is half the Euclidean distance from the best matching center, and a suitable overlap between adjacent rules is realized. As a result, the overlapping parameter setting is significant for the rule generation procedure.

B. Parameter Learning

Parameter learning is performed simultaneously to the structure-learning process. The gradient descent algorithm and variable-expansive Kalman filter algorithm are performed once each time that incoming data are derived. The antecedent parts and relative compensatory parameters in the TSCIT2FNN are adjusted for incoming data by the gradient descent algorithm, which is suitable for a supervised method. When we consider only the single-output case for clarity, the aim is to minimize the error function

$$E = \frac{1}{2}\left[y'(t+1) - y_d(t+1)\right]^2 \tag{19}$$

where $y'(t+1)$ and $y_d(t+1)$ represent the actual network output and the desired output, respectively. Next, we describe the consequent learning of details in the TSCIT2FNN based on the variable-expansive Kalman filter algorithm. In previous studies [42], the consequent part parameters must be reordered in ascending order, according to the KM iterative procedure, in order to find accurate end points, i.e., L and R points. Moreover, the design factor q enables to efficiently adjust the proportion of the upper and lower in the final output, and thus, we reduce the computational complexity as well as the problem of computation time without the KM iterative procedure in the TSCIT2FNN. The variable-expansive Kalman filter algorithm tunes the consequent part parameters. Therefore, (13) can be reexpressed as

$$y' = \vec{\phi}_{\text{TSK}}^T \vec{y}_{\text{TSK}} \tag{20}$$

where

$$\begin{aligned}&\vec{y}_{\text{TSK}} \\ &= \left[a_0^1, \ldots, a_n^1, \ldots, a_0^M, \ldots, a_n^M\right]^T\end{aligned} \tag{21}$$

$$\begin{aligned}&\vec{\phi}_{\text{TSK}}^T \\ &= \begin{bmatrix} \underbrace{q\underline{\phi}^1 \boldsymbol{x}_0 + (1-q)\overline{\phi}^1 x_0, \ldots, q\underline{\phi}^1 x_n + (1-q)\overline{\phi}^1 \boldsymbol{x}_n}_{n+1} \\ \vdots \\ \underbrace{q\underline{\phi}^M \boldsymbol{x}_0 + (1-q)\overline{\phi}^M x_0, \ldots, q\underline{\phi}^M x_n + (1-q)\overline{\phi}^M \boldsymbol{x}_n}_{n+1} \end{bmatrix}^T \\ &\in \Re^{1 \times M(n+1)}.\end{aligned} \tag{22}$$

Let $\underline{\phi}$ and $\overline{\phi}$ be defined as follows:

$$\underline{\phi} = \frac{\underline{\psi}}{\sum_{i=1}^{M} \underline{\psi}^i(t)}, \quad \overline{\phi} = \frac{\overline{\psi}}{\sum_{i=1}^{M} \overline{\psi}^i(t)}. \tag{23}$$

The consequent parameter vector is updated by executing the following variable-expansive Kalman filter algorithm:

$$\begin{aligned}&\vec{y}_{\text{TSK}}(t+1) \\ &\quad = \vec{y}_{\text{TSK}}(t) + S(t+1)\vec{\phi}_{\text{TSK}}(t+1)\end{aligned}$$

$$\times\left(y_d(t+1)-\vec{\phi}^T_{\rm TSK}(t+1)\vec{y}_{\rm TSK}(t)\right)$$

$$S(t+1)=\frac{1}{\lambda}\left[S(t)-\frac{S(t)\vec{\phi}_{\rm TSK}(t+1)\vec{\phi}^T_{\rm TSK}(t+1)S(t)}{\lambda+\vec{\phi}^T_{\rm TSK}(t+1)S(t)\vec{\phi}_{\rm TSK}(t+1)}\right] \quad (24)$$

where λ is a forgetting factor $(0<\lambda\le 1)$. The dimension of the vectors $\vec{y}_{\rm TSK}$ and $\vec{\phi}_{\rm TSK}$, and the matrix S, increases whenever a new rule evolves. When a new rule evolves, we express the argument $S(t)$ in the TSCIT2FNN as

$$S(t)=\text{block diag}\,[S(t),C\cdot I]\in\Re^{(M+1)(n+1)\times(M+1)(n+1)} \quad (25)$$

where C is a large positive constant. The important problem in this algorithm is theoretical convergence. Many researchers have reported on and analyzed [52], [53] the Kalman filter algorithm with a fixed input dimension. Moreover, the theoretical convergence analysis concerning varying input dimensions is an issue for another investigation. In order to experimentally address this problem, the matrix S must be reset to $C\cdot I$. The aim of this resetting operation is to keep S bounded.

We now describe the antecedent parameter and compensatory coefficient learning of the TSCIT2FNN based on the gradient descent algorithm as follows.

Let w_j^i denote the antecedent parameters, m_1, m_2, and σ, in the ith interval type-2 set in input variable x_j

$$w_j^i(t+1)=w_j^i(t)-\eta\frac{\partial E}{\partial w_j^i(t)} \quad (26)$$

where

$$\frac{\partial E}{\partial w_j^i}=(y'-y_d)\left[\left(\frac{\partial y'}{\partial \underline{\psi}^i}\frac{\partial \underline{\psi}^i}{\partial \underline{f}^i}\frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial w_j^i}\right)+\left(\frac{\partial y'}{\partial \overline{\psi}^i}\frac{\partial \overline{\psi}^i}{\partial \overline{f}^i}\frac{\partial \overline{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial w_j^i}\right)\right] \quad (27)$$

and η is a learning coefficient in the range [0,1].

$$\frac{\partial y'}{\partial \underline{\psi}^i}=q\frac{y^i_{\rm TSK}-\underline{y}'}{\sum_{i=1}^M \underline{\psi}^i}\quad \frac{\partial y'}{\partial \overline{\psi}^i}=(1-q)\frac{y^i_{\rm TSK}-\overline{y}'}{\sum_{i=1}^M \overline{\psi}^i}$$

$$y'=q\underline{y}'+(1-q)\overline{y}'=q\frac{\sum_{i=1}^M \underline{\psi}^i y^i_{\rm TSK}}{\sum_{i=1}^M \underline{\psi}^i}+(1-q)\frac{\sum_{i=1}^M \overline{\psi}^i y^i_{\rm TSK}}{\sum_{i=1}^M \overline{\psi}^i}. \quad (28)$$

In (28), $y^i_{\rm TSK}$ represents the consequent value, which can be seen in (12).

If $w_j^i=m_{j1}^i$, then we have

$$\frac{\partial \underline{\psi}^i}{\partial m_{j1}^i}=\frac{\partial \underline{\psi}^i}{\partial \underline{f}^i}\frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j1}^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\underline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\underline{f}^i)\left(\frac{x_j-m_{j1}^i}{(\sigma_j^i)^2}\right), & x_j>\frac{m_{j1}^i+m_{j2}^i}{2}\\ 0, & \text{otherwise}\end{cases} \quad (29)$$

$$\frac{\partial \overline{\psi}^i}{\partial m_{j1}^i}=\frac{\partial \overline{\psi}^i}{\partial \overline{f}^i}\frac{\partial \overline{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j1}^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\overline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\overline{f}^i)\left(\frac{x_j-m_{j1}^i}{(\sigma_j^i)^2}\right), & x_j\le m_{j1}^i\\ 0, & \text{otherwise.}\end{cases} \quad (30)$$

Similarly, if $w_j^i=m_{j2}^i$, then we have

$$\frac{\partial \underline{\psi}^i}{\partial m_{j2}^i}=\frac{\partial \underline{\psi}^i}{\partial \underline{f}^i}\frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial m_{j2}^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)\\ \quad\times(\underline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}(\underline{f}^i)\left(\frac{x_j-m_{j2}^i}{(\sigma_j^i)^2}\right), & x_j\le\frac{m_{j1}^i+m_{j2}^i}{2}\\ 0, & \text{otherwise}\end{cases} \quad (31)$$

$$\frac{\partial \overline{\psi}^i}{\partial m_{j2}^i}=\frac{\partial \overline{\psi}^i}{\partial \overline{f}^i}\frac{\partial \overline{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial m_{j2}^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\overline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\overline{f}^i)\left(\frac{x_j-m_{j2}^i}{(\sigma_j^i)^2}\right), & x_j>m_{j1}^i\\ 0, & \text{otherwise.}\end{cases} \quad (32)$$

If $w_j^i=\sigma_j^i$, then we have

$$\frac{\partial \underline{\psi}^i}{\partial \sigma_j^i}=\frac{\partial \underline{\psi}^i}{\partial \underline{f}^i}\frac{\partial \underline{f}^i}{\partial \underline{\mu}_j^i}\frac{\partial \underline{\mu}_j^i}{\partial \sigma_j^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\underline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\underline{f}^i)\frac{(x_j-m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j\le\frac{m_{j1}^i+m_{j2}^i}{2}\\ \left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\underline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\underline{f}^i)\frac{(x_j-m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j>\frac{m_{j1}^i+m_{j2}^i}{2}\end{cases} \quad (33)$$

$$\frac{\partial \overline{\psi}^i}{\partial \sigma_j^i}=\frac{\partial \overline{\psi}^i}{\partial \overline{f}^i}\frac{\partial \overline{f}^i}{\partial \overline{\mu}_j^i}\frac{\partial \overline{\mu}_j^i}{\partial \sigma_j^i}=\begin{cases}\left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\overline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\overline{f}^i)\frac{(x_j-m_{j1}^i)^2}{(\sigma_j^i)^3}, & x_j<m_{j1}^i\\ \left(1-\gamma^i+\frac{\gamma^i}{n}\right)(\overline{f}^i)^{-\gamma^i+\frac{\gamma^i}{n}}\\ \quad\times(\overline{f}^i)\frac{(x_j-m_{j2}^i)^2}{(\sigma_j^i)^3}, & x_j>m_{j2}^i\\ 0, & \text{otherwise.}\end{cases} \quad (34)$$

As mentioned earlier, the design factor q allows us to adjust the lower and upper portions in the final output according to the following formulas:

$$q(t+1)=q(t)-\eta\frac{\partial E}{\partial q}$$

$$\frac{\partial E}{\partial q}=(y'-y_d)\left(\frac{\sum_{i=1}^M \underline{\psi}^i y^i_{\rm TSK}}{\sum_{i=1}^M \underline{\psi}^i}-\frac{\sum_{i=1}^M \overline{\psi}^i y^i_{\rm TSK}}{\sum_{i-1}^M \overline{\psi}^i}\right). \quad (35)$$

Next, we describe the relative compensatory coefficient γ^i that is updated by the gradient descent algorithm

$$\frac{\partial E}{\partial \gamma^i} = \frac{\partial E}{\partial y'}\left[\left(\frac{\partial y'}{\partial \underline{\psi}^i}\frac{\partial \underline{\psi}^i}{\partial \gamma^i}\right) + \left(\frac{\partial y'}{\partial \overline{\psi}^i}\frac{\partial \overline{\psi}^i}{\partial \gamma^i}\right)\right]$$
$$= \frac{\partial E}{\partial y'}\left[\left(\frac{\partial y'}{\partial \underline{\psi}^i}\right)\left(\ln|\underline{f}^i|\right)(\underline{\psi}^i)\left(\frac{1}{n}-1\right) + \left(\frac{\partial y'}{\partial \overline{\psi}^i}\right)\left(\ln|\overline{f}^i|\right)(\overline{\psi}^i)\left(\frac{1}{n}-1\right)\right]. \quad (36)$$

Then, the parameters c_i and d_i are the relative coefficients of the compensatory degree γ^i

$$c_i(t+1) = c_i(t) + \eta_c\left\{\frac{2c_i(t)d_i^2(t)}{[c_i^2(t)+(d_i^2(t)]^2}\right\}\frac{\partial E}{\partial \gamma^i} \quad (37)$$

$$d_i(t+1) = d_i(t) - \eta_d\left\{\frac{2c_i^2(t)d_i(t)}{[c_i^2(t)+(d_i^2(t)]^2}\right\}\frac{\partial E}{\partial \gamma^i} \quad (38)$$

$$\gamma^i(t+1) = \frac{c_i^2(t+1)}{c_i^2(t+1)+d_i^2(t+1)} \quad (39)$$

where the different learning rates, i.e., η_c and η_d, enable us to adapt the compensatory degree more efficiently.

The gradient descent algorithm is used in a TSCIT2FNN for online learning, although it may have defective characteristics in terms of learning convergence and the local minimum problem. Convergence is guaranteed by the following condition:

$$0 < \eta(t) < \frac{2}{\left(\max_t \left\|\frac{\partial y'(t)}{\partial W}\right\|\right)}.$$

The derivation of this condition can be found in [48] and [65] and is given in the Appendix.

The learning rate η is within the range [0, 1]. A large value results in extreme oscillations of the updated parameters. On the other hand, a small learning rate makes the learning convergence slower. It is important that the proper learning rate be chosen to adapt the system.

IV. Simulations

This section describes five examples to evaluate the performance of the TSCIT2FNN. These simulation studies include three types of system identification, ANC, and prediction of the Mackey–Glass time series. As a result, the performance of our TSCIT2FNN is compared with that of other existing type-1 and type-2 FNNs.

A. Example 1 (System Identification)

The TSCIT2FNN is applied to the identification of a nonlinear system, which has been presented in [3]. The process is described by the difference equation

$$y_d(t+1) = \frac{y_d(t)}{1+y_d^2(t)} + u^3(t). \quad (40)$$

The training signal is generated with $u(t) = \sin(2\pi t/100)$, $t = 1, \ldots, 200$. The TSCIT2FNN inputs are $u(t)$ and $y_d(t)$, and the desired output is $y_d(t+1)$. As a performance criterion, we evaluate the root-mean-square error (rmse)

$$rmse = \sqrt{\frac{1}{200}\sum_{t=1}^{200}[y'(t+1) - y_d(t+1)]} \quad (41)$$

where $y'(t+1)$ is the TSCIT2FNN output. The learning rate η is set to 0.075. The structure threshold f_{th} determines the number of rules. When f_{th} is set to 0.02, three fuzzy rules are constructed. Training is performed for 500 epochs with 200 steps. After training, the rmse obtained is 0.0078. All networks use the identical training, test data, and number of training epochs as the TSCIT2FNN. Fig. 3 shows the relative coefficients of a compensatory operation during training epochs. Fig. 4 shows the TSCIT2FNN results for the test.

This example also computes the performance of TSCIT2FNN with different models as shown in Table I. Table I shows the network performance, including the network size, training rmse, and test rmse, and compares them with other type-1 and type-2 FNNs presented in [3], [44], and [48].

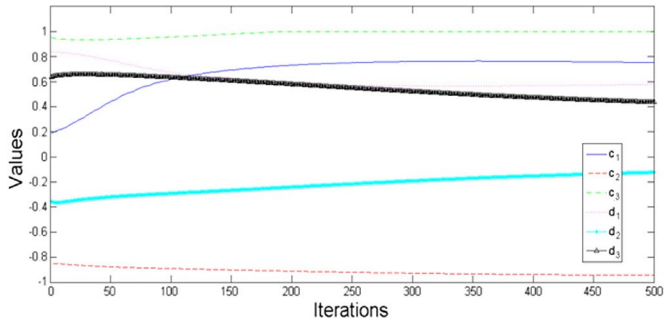

Fig. 3. Simulation curve of the relative weights of compensatory operation during learning.

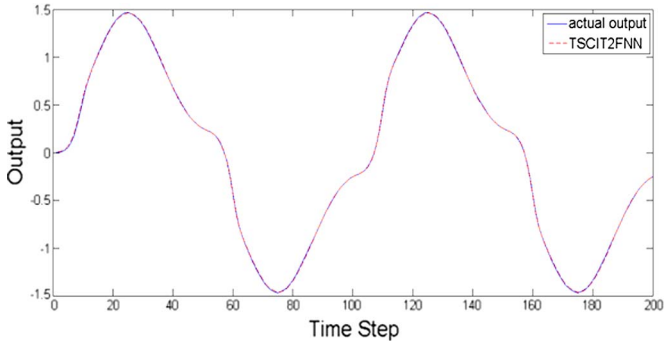

Fig. 4. Identification result of the TSCIT2FNN.

TABLE I
Performance Comparison for Example 1

Models	Rules	Parameter number	Test RMSE
SONFIN[3]	7	35	0.0085
Type-2 TSK FLS [44]	3	36	0.0388
Feedforward Type-2 FNN	3	36	0.0281
T2 TSK FNS [48]	4	24	0.0324
SEIT2FNN [44]	3	36	0.0062
TSCIT2FNN	3	34	0.0084

The feedforward type-1 FNN that we use for comparison is a SONFIN [3], a powerful network with both structure and parameter learning abilities. The consequent part of the SONFIN is of the Mamdani type. All free parameters in SONFIN are learned using the gradient descent algorithm. The original interval type-2 FNN for comparison is presented in [39], where the type-2 FLS structure is fixed and assigned in advance. In an attempt to have similar network size for comparison, the total number of parameters in a feedforward type-1 FNN is designed to be close to that in a feedforward type-2 FNN.

The type-2 TSK FNS [48] has fewer parameters than other type-2 FNNs because the type-2 TSK FNS uses a crisp linear function in the consequent part, as does the TSCIT2FNN. The feedforward type-2 FNN, being of the TSK type in the consequent part, is similar to the SEIT2FNN structure because they use simultaneous structure and parameter learning online. However, the feedforward type-2 FNN uses a gradient descent algorithm to tune the free parameters. As can be seen in Table I, the performance of the feedforward type-2 FNN is inferior to that of the SEIT2FNN using gradient descent and Kalman filter algorithms. The type-2 TSK FLS must define rules in advance and uses the FCM to construct rules. For parameter learning, the type-2 TSK FLS and type-2 TSK FNS use the gradient descent algorithm to achieve the optimal values. In comparison with the type-2 models, the SEIT2FNN achieves a smaller test rmse than the other models. This is because the SEIT2FNN has distinguishing characteristics for obtaining a better performance, including a self-evolving ability and the use of a rule-ordered Kalman filter algorithm to tune interval weights in the consequent part.

In general, the type-reduction process in the feedforward type-2 models, i.e., the type-2 TSK FLS and the SEIT2FNN [44], is more difficult and time consuming. We now analyze the practical computational cost of constructing a TSCIT2FNN. All simulations are performed on an Intel 2.53-GHz dual central processing unit, and the programs are written in Visual C++. The execution time for the learning process of the TSCIT2FNN is 1.14 s. The learning times of three representative models, SONFIN, type-2 TSK FLS, and SEIT2FNN, take 0.7, 1.55, and 1.42 s, respectively. The results show that the computational cost of our TSCIT2FNN is less than that of two type-2 models.

We now compare the performance of SEIT2FNN and type-2 TSK FNS with our approach. The test error of TSCIT2FNN is close to that of SEIT2FNN but has fewer parameters. In contrast to the type-2 TSK FNS [48], our network yields a lower test rmse, as can be seen in Table I. Both TSCIT2FNN and type-2 TSK FNS obtain their final output based on design factor q. The results show that the TSCIT2FNN rmse is smaller than that of the other type-2 models, except the SEIT2FNN.

For the further simulation studies, the modification plant is described by

$$y_d(t+1) = \frac{\alpha(t) y_d(t)}{1 + y_d^2(t)} + u^3(t) \tag{42}$$

where

$$\alpha(t) = \begin{cases} 1, & 1 \le t \le 1000 \\ \sin(t/100), & t > 1000. \end{cases} \tag{43}$$

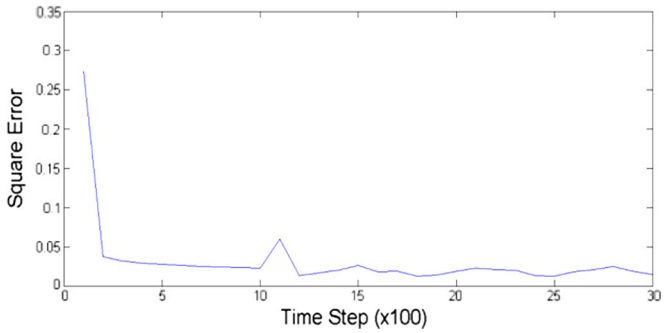

Fig. 5. RMSE values obtained from online learning.

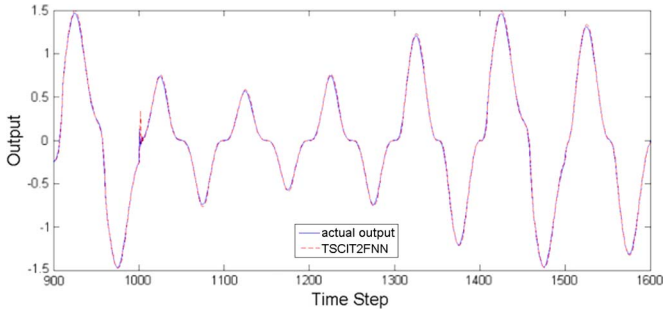

Fig. 6. Online identification results of the actual output and the TSCIT2FNN.

The signal $u(t) = \sin(2\pi t/100)$ is applied to the system in (42) as input, and then, three rules are generated after 1000 time steps of online learning. The time-varying system continuously changes after 1000 time steps, and additional five rules are generated and cover the following new inputs. Fig. 5 shows the TSCIT2FNN rmses that are accumulated over 100 adjacent time steps. Fig. 6 shows the learning result of the actual output and the TSCIT2FNN. The results demonstrate the TSCIT2FNN can efficiently handle time-varying system with better performance.

B. Example 2 (Second-Order System Identification)

This example uses the TSCIT2FNN to identify a second-order time-varying system, a problem that was introduced in [48] and [64]. The dynamic system is described by the following difference equation:

$$y_p(t+1) = f\left(y_p(t), y_p(t-1), y_p(t-2), u(t), u(t-1)\right) \tag{44}$$

where

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{x_1 x_2 x_3 x_5 (x_3 - b) + c x_4}{a + x_2^2 + x_3^2}. \tag{45}$$

The time-varying parameters $a(t)$, $b(t)$, and $c(t)$ are given by

$$a(t) = 1.2 - 0.2\cos(2\pi t/T) \tag{46}$$

$$b(t) = 1.0 - 0.4\sin(2\pi t/T) \tag{47}$$

$$c(t) = 1.0 + 0.4\sin(2\pi t/T) \tag{48}$$

where T is the maximum time step.

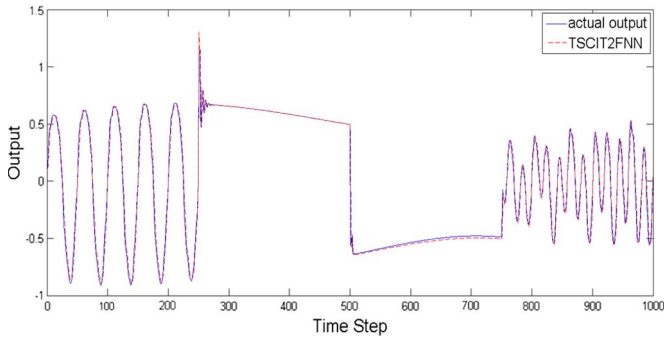

Fig. 7. Identification result of the TSCIT2FNN using three rules.

TABLE II
PERFORMANCE COMPARISON FOR EXAMPLE 2

Models	Rules	Epochs	Training RMSE	Test RMSE
T1 TSK FNS [48]	9	100	0.0282	0.0598
T2 TSK FNS [48]	4	100	0.0284	0.0601
Feedforward Type-2 FNN	3	100	0.0281	0.0593
SEIT2FNN [44]	3	100	0.0274	0.0574
TSCIT2FNN	3	100	0.0279	0.0576

The dynamic system output depends on three previous outputs and one previous input. In this example, only two current values, $u(t)$ and $y_p(t)$, are fed as input to the TSCIT2FNN input layer. In the training procedure of the TSCIT2FNN, we follow the same computational protocols as in [48]. After 100 epochs of training, three rules are generated when the structure-learning threshold is set to 0.04. The learning coefficient is set to 0.07. To evaluate the identification result, the following signal is used for the test:

$$u(t)=\begin{cases}\sin\left(\frac{\pi t}{25}\right), & t<250\\ 1.0, & 250\le t<500\\ -1.0, & 500\le t<750\\ 0.3\sin\left(\frac{\pi t}{25}\right)+0.1\sin\left(\frac{\pi t}{32}\right) & \\ \quad +0.6\sin\left(\frac{\pi t}{10}\right), & 750\le t<1000.\end{cases} \tag{49}$$

Fig. 7 shows the outputs of the plant and the TSCIT2FNN for these test inputs.

The three models, type-1 and type-2 TSK FNSs and the SEIT2FNN, use an identical number of input variables, training data, test data, and training epochs as the TSCIT2FNN. For comparison with test rmse, they are almost identical. Neither the type-2 TSK FNS nor the SEIT2FNN use the compensatory operation. The parameter learning of the type-1 TSK FNS is trained using gradient descent-based optimization. Although the feedforward models were unable to effectively handle dynamic systems, the results in Table II indicate that the TSCIT2FNN achieves similar performance with fewer rules.

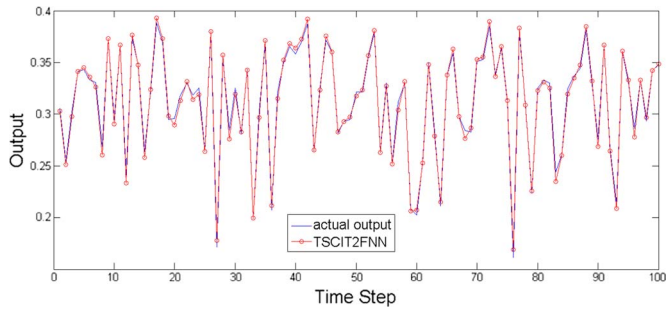

Fig. 8. Simulation curve of the TSCIT2FNN output.

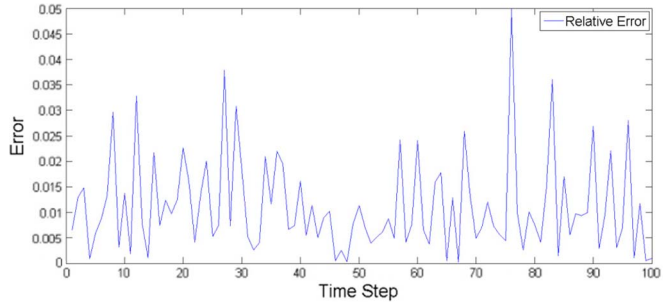

Fig. 9. Simulation curve of absolute relative error between actual output and the TSCIT2FNN.

TABLE III
PERFORMANCE COMPARISON FOR EXAMPLE 3

Models	Max. (%) relative error	Min. (%) relative error	Avg. (%) relative error
Fuzzy-rules RBFNN [54]	11.53	0.0024	3.18
ENN [55]	13.05	0. 0519	3.82
RFNN [56]	6.29	0. 0328	2.91
WFNN [12]	3.84	0. 0152	1.79
Feedforward Type-2 FNN	6.01	0.0318	1.42
SEIT2FNN [44]	8.25	0. 0307	1.02
TSCIT2FNN	4.85	0. 0201	1.12

C. Example 3 (Nonlinear Function Identification)

To assess the validity of the TSCIT2FNN, a data synthesis is made with the following function:

$$y=\sqrt{64-81\cdot((x_1-0.6)^2+(x_2-0.5)^2}/9-0.5. \tag{50}$$

This plant has two inputs (x_1 and x_2) and one output y, so the two input variables are fed as inputs to the TSCIT2FNN input layer. As in [54], the inputs are randomly generated within [0, 1]. To generate 1100 patterns, the first 1000 patterns are used for training, and the remaining 100 patterns are used for testing. The structure threshold f_{th}, learning rate η, and design factor q are set to 0.05, 0.075, and 0.3, respectively. After 100 epochs of training, three rules are generated. Fig. 8 depicts the actual output and the TSCIT2FNN output response for the test patterns. The absolute relative error of the TSCIT2FNN is also shown in Fig. 9. Table III shows the maximum, minimum, and average relative error. The average relative error of the

TSCIT2FNN is less than 2%. We also compare the performance of the SEIT2FNN [44] and TSCIT2FNN. As can be seen from Table III, the average relative error of the TSCIT2FNN is similar to that of the SEIT2FNN. Indeed, the results in Table III reveal the advantage of the TSCIT2FNN. As mentioned earlier, the combination of technologies in the TSCIT2FNN, like compensatory operation and the use of a design factor to adjust portions of the lower and upper, provides more accurate results, which is not the case with SEIT2FNN.

Table III compares the performance of the TSCIT2FNN with that of existing type-1 FNNs, including an Elman-style recurrent network (ENN) [55], a fuzzy-rule radial basis function neural network [54], a recurrent FNN (RFNN) [56], and a wavelet FNN (WFNN) [12]. These models used an identical number of training epochs, rules, and training and test samples as the TSCIT2FNN. Of the type-1 FNNs, the WFNN achieves the smallest relative error. This is because the consequent part of the WFNN consists of wavelet basis functions, which have the ability to localize both in the time and frequency domains. The results indicate that the TSCIT2FNN obtains better performance than the other networks.

D. Example 4 (ANC)

In this example, we consider noise tolerance and system recovery. The TSCIT2FNN is applied to the ANC problem presented in [57] and [58]. ANC is concerned with the enhancement of noise corrupted signals and has been used successfully in various areas, such as interference cancellation in electrocardiograms, echo elimination on long-distance telephone transmission lines, and antenna interference canceling.

ANC is based on the availability of a primary input source and an auxiliary (reference) input source located at the noise field that contains little or none, as shown in Fig. 10. In this figure, the primary input source contains the desired output s and the noise n_0 generated from the noise source n in order to corrupt the desired output. The received signal is thus

$$x(t) = s(t) + n_0(t). \tag{51}$$

The secondary or auxiliary (reference) input source receives the noise n_1, which is correlated with the corrupting noise n_0. The purpose of ANC techniques is to adaptively process the reference noise n_0, which is generated as a replica of noise n_1, and then subtract this from the primary input x to recover the desired signal s. The TSCIT2FNN output y represents a replica of n_0. In [58], the assumptions are made that s, n_0, and n_1 are stationary zero-mean processes, s is independent, and n_0 and n_1 are dependent. Therefore, the reference input source is situated in such a position that it detects only the noise but not the signal s. From Fig. 10, we can suppose that

$$e(t) = s(t) + n_0(t) - y(t) \tag{52}$$

where

$$s(t) = 0.6\sin(0.06t)\cos(0.01t) \tag{53}$$

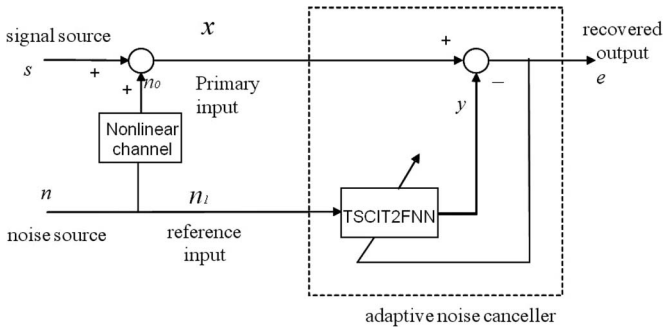

Fig. 10. ANC using the TSCIT2FNN.

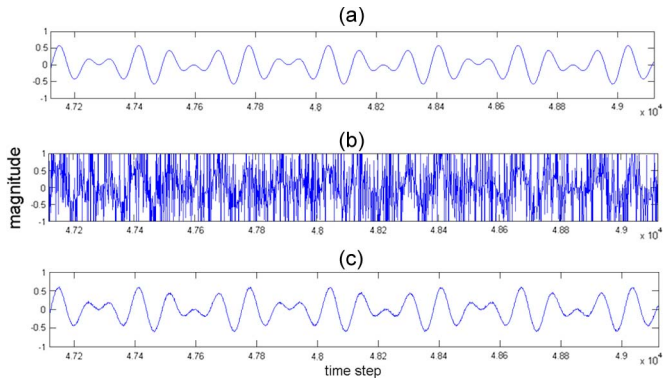

Fig. 11. (a) Original signal source s. (b) Corrupted signal x. (c) Recovered signal e.

and the noise source n is white noise with a uniform distribution in $[-1.5, 1.5]$. The relation between the noise source n and the corrupting noise n_1 is a nonlinear function that is expressed as

$$n_0 = 0.6\,(n(t))^3\,. \tag{54}$$

The reference input is placed in front of the noise source so that $n = n_1$. The reference input n_1 is fed as input, and then, the actual output of the TSCIT2FNN is derived. The learning coefficient η is set to 0.01. The TSCIT2FNN generates seven rules during the online training process when the structure-learning threshold f_{th} is set to 0.5. Fig. 11 shows the detailed training process from time step 48 000 to 50 000, where Fig. 11(a)–(c) shows the original, corrupted, and recovered signals, respectively. The performance of the TSCIT2FNN is compared to that of the adaptive neural fuzzy filter (AFNN) [57] and the SEIT2FNN [44]. The AFNN is a type-1 fuzzy structure with both structure and parameter learning abilities. Both AFNN and SEIT2FNN generate seven rules during the online training process.

The square errors between the recovered and original signals are computed such that each plotted error value is the sum of square errors over 60 adjacent time steps for each network, as shown in Fig. 12. This figure illustrates that the TSCIT2FNN has faster convergence than the other models. The result shows that the two type-2 structures, i.e., SEIT2FNN and TSCIT2FNN, are better than the type-1 structure (AFNN) with identical fuzzy rules. Fig. 12 illustrates that the TSK-type models with interval weights in the consequent part are susceptible to noisy environments.

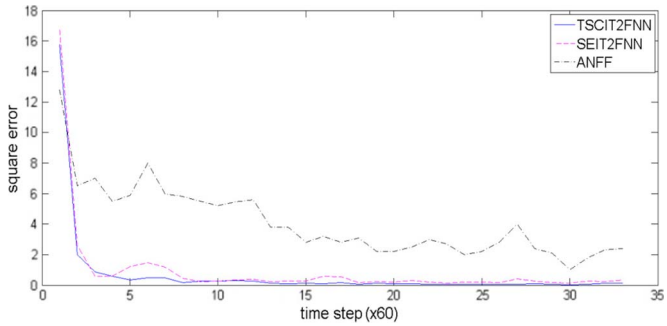

Fig. 12. Square error compared with the different models.

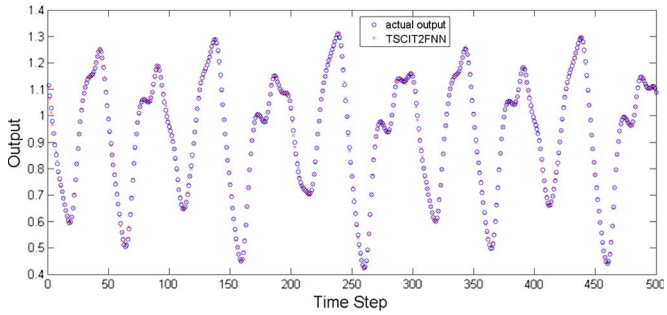

Fig. 13. Prediction results using the TSCIT2FNN.

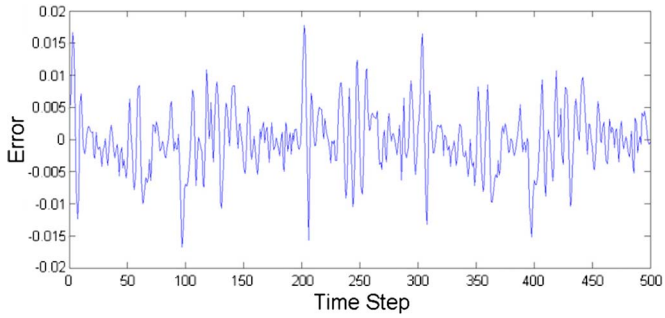

Fig. 14. Test error of the chaotic time-series prediction using the TSCIT2FNN.

TABLE IV
PERFORMANCE COMPARISON FOR EXAMPLE 5

Models	Rules	Test RMSE
RBF-AFS [59]	21	0.013
HyFIS [60]	16	0.012
NEFPROX [61]	26	0.053
D-FNN [62]	10	0.008
FWNN-R [11]	16	0.002
LLWNN+hybrid [63]	-	0.004
Feedforward type-2 FNN	7	0.006
SETI2FNN [44]	7	0.003
TSCIT2FNN	7	0.005

E. Example 5 (Prediction of a Chaotic Time Series)

This example considers a prediction problem that has been used in [44] and [59]–[63]. The proposed TSCIT2FNN model is applied to the Mackey–Glass time series that is generated from the following delay differential equation:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t). \tag{55}$$

For $\tau = 17$, the system response is chaotic, and simulation data are obtained using the initial conditions $x(0) = 1.2$ and $\tau = 17$, as in the previous literature. We generate 1000 input–output data points from $t = 124$ to 1123, with the first 500 patterns being used for training and the remaining 500 for testing. Four past values are used to predict $x(t)$, and the input–output data format is $[x(t-24), x(t-18), x(t-12), x(t-6); x(t)]$.

The structure-learning threshold f_{th}, learning factor η, and design factor q are set to 0.2, 0.075, and 0.3, respectively. After 500 training epochs, seven rules are generated. Fig. 13 shows the prediction result of the TSCIT2FNN. Fig. 14 depicts the prediction error between the actual result and that of the TSCIT2FNN, which shows that there is an excellent match between them.

A comparison of the TSCIT2FNN model with different models, including the radial based adaptive fuzzy system (RBF-AFS) [59], hybrid neural fuzzy inference system (HyFIS) [60], a neuro-fuzzy function approximator (NEFPROX) [61], dynamic FNN (DFNN) [62], FWNN [11], local linear wavelet neural network with hybrid model (LLWNN+hybrid) [63], and SEIT2FNN [44], is illustrated in Table IV. For a fair comparison, the proposed approach is compared with SEIT2FNN, which also uses a type-2 fuzzy set with uncertain means in the antecedent part. The result indicates that the SEIT2FNN performs better than the proposed approach. The reason for this is that the SEIT2FNN achieves better performance in the consequent parts using interval type-1 sets. However, the proposed approach has fewer parameters than the SEIT2FNN and is thus less computationally expensive.

The performance of the TSCIT2FNN is also comparable to that of recently developed type-1 FNNs. As can be seen from Table IV, the FWNN is one of the best models in terms of low test rmse. For their consequent parts, both the FWNN and the LLWNN are composed of wavelet basis functions, which have the ability to localize both in the time and frequency domains. A hybrid algorithm of particle swarm optimization and gradient descent is used for training the LLWNN. A fast gradient-based training algorithm, the Broyden–Fletcher–Goldfarb–Shanno method, is used to find optimal values for free parameters of the FWNN models. However, the performance of the TSCIT2FNN is close to the FWNN's performance, due in part to having fewer rules. Finally, the results indicate that the TSCIT2FNN achieves a smaller prediction error and has fewer rules than type-1 fuzzy models.

We now analyze the practical computational cost of constructing a TSCIT2FNN. The execution time for the learning process of the TSCIT2FNN is 9.26 s. The SEIT2FNN takes 37.3 s. The results show that the computational cost of the TSCIT2FNN is much less than that of the SEIT2FNN.

V. CONCLUSION

In this paper, a TSK-type-based compensatory interval type-2 FNN with a self-evolving structure and parameter learning was proposed. The FoU of type-2 fuzzy sets in the

TSCIT2FNN enables the system to handle the numerical uncertainty as well as uncertain information. To obtain the optimal type-2 fuzzy logic reasoning and select the optimal type-2 fuzzy operation, we have exploited a compensatory operator. Thus, our TSCIT2FNN can adaptively tune the type-2 fuzzy MF and efficiently optimize the fuzzy operator. In terms of structure learning, there is no need to decide the TSCIT2FNN rules in advance. The TSCIT2FNN provides the self-evolving ability to structure itself online, which is especially useful for handling industrial problems with temporally varying characteristics. The premise and consequent parameters are learned as gradient descent and variable-expansive Kalman filter algorithms to improve the learning accuracy. Finally, the TSCIT2FNN has potential to deal with noisy data, as verified in the simulation results. In summary, the TSCIT2FNN variance system achieved a better performance than existing type-1 or type-2 FNNs in identification, prediction, and ANC.

APPENDIX

This appendix derives the optimal learning rate using a Lyapunov function by following the previous studies [48], [65]. Let $\eta(t)$ be the learning rate at discrete time t for the update formula of the TSCIT2FNN weights. The convergence is guaranteed if $\eta(t)$ is satisfied by the following condition:

$$0 < \eta(t) < \frac{2}{\left(\max\limits_{t} \left\| \frac{\partial y'(t)}{\partial W} \right\|\right)}. \tag{A1}$$

The statement in (A1) can be proved by choosing the following Lyapunov function:

$$V(t) = \frac{1}{2}e^2(t), \quad e(t) = (y_d(t) - y'(t)). \tag{A2}$$

The change in the Lyapunov function is obtained as

$$\begin{aligned}\Delta V(t) &= V(t+1) - V(t) = \frac{1}{2}\left(e^2(t+1) - e^2(t)\right) \\ &= \frac{1}{2}\left((e(t) + \Delta e(t))^2 - e^2(t)\right) \\ &= \frac{1}{2}\left(2e(t)\cdot\Delta e(t) + \Delta e^2(t)\right) \\ &= \frac{1}{2}\Delta e(t)\left(2e(t) + \Delta e(t)\right). \end{aligned} \tag{A3}$$

The error difference can be represented as

$$\Delta e(t) = \left(\frac{\partial e(t)}{\partial W}\right)^T$$

$$\Delta W = \left[\frac{\partial\left(y_d(t) - y'(t)\right)}{\partial W}\right]^T$$

$$\Delta W = -\left[\frac{\partial y'(t)}{\partial W}\right]\Delta W. \tag{A4}$$

From the update formulas, we get

$$\Delta W = -\eta\frac{\partial E}{\partial W} = \eta e(t)\frac{\partial y'(t)}{\partial W}$$

where

$$\frac{\partial E}{\partial W} = \frac{\partial}{\partial W}\left[\frac{1}{2}e^2(t)\right] = e(t)\frac{\partial e(t)}{\partial W} = -e(t)\frac{\partial y'(t)}{\partial W} \tag{A5}$$

$$\begin{aligned}\Delta V(t) &= \frac{1}{2}\Delta e(t)\left(2e(t) + \Delta e(t)\right) \\ &= \frac{1}{2}\eta(t)e^2(t)\left\|\frac{\partial y'(t)}{\partial W}\right\|^2\left(\eta(t)\left\|\frac{\partial y'(t)}{\partial W}\right\|^2 - 2\right). \end{aligned} \tag{A6}$$

From the Lyapunov stability theorem, asymptotic stability is guaranteed under the following sufficient condition:

$$0 < \eta(t) < \frac{2}{\left(\max\limits_{t}\left\|\frac{\partial y'(t)}{\partial W}\right\|\right)}. \tag{A7}$$

REFERENCES

[1] L. A. Zadeh, "The concept of linguistic variable and its application to approximate reasoning," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975.

[2] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1997, ch. 17.

[3] C. F. Juang and C. T. Lin, "An on-line self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, Feb. 1998.

[4] R. J. Wai and P. C. Chen, "Intelligent tracking control for robot manipulator including actuator dynamics via TSK-type fuzzy neural network," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 552–560, Aug. 2004.

[5] Y. X. Liao, J. H. She, and M. Wu, "Integrated hybrid-PSO and fuzzy-NN decoupling control for temperature of reheating furnace," *IEEE Trans. Ind. Electron.*, vol. 56, no. 7, pp. 2704–2714, Jul. 2009.

[6] C. T. Lin, C. M. Yeh, S. F. Liang, J. F. Chung, and N. Kumar, "Support-vector-based fuzzy neural network for pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 1, pp. 31–41, Feb. 2006.

[7] L. Y. Shyu, Y. H. Wu, and W. Hu, "Using wavelet transform and fuzzy neural network for VPC detection from the holter ECG," *IEEE Trans. Biomed. Eng.*, vol. 51, no. 7, pp. 1269–1273, Jul. 2004.

[8] I. S. Li, W. Y. Wang, S. F. Su, and Y. S. Lee, "A merged fuzzy neural network and its applications in battery state-of-charge estimation," *IEEE Trans. Energy Convers.*, vol. 22, no. 3, pp. 697–708, Sep. 2007.

[9] A. Rubaai, A. R. Ofoli, L. Burge, III, and M. Garuba, "Hardware implementation of an adaptive network-based fuzzy controller for dc-dc converters," *IEEE Trans. Ind. Appl.*, vol. 41, no. 6, pp. 1557–1565, Nov./Dec. 2005.

[10] R. H. Abiyev and O. Kaynak, "Fuzzy wavelet neural networks for identification and control of dynamic plants—A novel structure and a comparative study," *IEEE Trans. Ind. Electron.*, vol. 55, no. 8, pp. 3133–3140, Aug. 2008.

[11] S. Yilmaz and Y. Oysal, "Fuzzy wavelet neural network models for prediction and identification of dynamical systems," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1599–1609, Oct. 2010.

[12] C. H. Lu, "Wavelet fuzzy neural network for identification and predictive control of dynamic systems," *IEEE Trans. Ind. Electron.*, vol. 58, no. 7, pp. 3046–3058, Jul. 2011.

[13] H. J. Zimmermann and P. Zysno, "Latent connective in human decision making," *Fuzzy Sets Syst.*, vol. 4, no. 1, pp. 37–51, Jul. 1980.

[14] Y. Q. Zhang and A. Kandel, "Compensatory neurofuzzy systems with fast learning algorithm," *IEEE Trans. Neural Netw.*, vol. 9, no. 1, pp. 83–105, Jan. 1998.

[15] S. Halgamuge and M. Glesner, "Neural networks in designing fuzzy systems for real world application," *Fuzzy Sets Syst.*, vol. 65, no. 1, pp. 1–12, Jul. 1994.

[16] N. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid system," *Fuzzy Sets Syst.*, vol. 82, no. 2, pp. 135–149, Sep. 1996.
[17] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from date," *Fuzzy Sets Syst.*, vol. 89, no. 3, pp. 277–288, Aug. 1997.
[18] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 578–599, May 2002.
[19] J. S. Wang and G. C. S. Lee, "Self-adaptive neuro-fuzzy inference system for classification applications," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 6, pp. 790–802, Dec. 2002.
[20] H. Seker, D. E. Evans, N. Aydin, and E. Yazgan, "Compensatory fuzzy neural networks-based intelligent detection of abnormal neonatal cerebral Doppler ultrasound waveforms," *IEEE Trans. Inf. Technol. Biomed.*, vol. 5, no. 3, pp. 187–194, Sep. 2001.
[21] C. J. Lin and W. H. Ho, "A pseudo-Gaussian-based compensatory neural fuzzy system," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, May 25–28, 2003, pp. 214–219.
[22] C. J. Lin and C. H. Chen, "Nonlinear system control using compensatory neuro-fuzzy networks," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E86-A, no. 9, pp. 2309–2316, 2003.
[23] C. H. Chen, C. J. Lin, and C. T. Lin, "An efficient quantum neuro-fuzzy classifier based on fuzzy entropy and compensatory operation," *Soft Comput.*, vol. 12, no. 6, pp. 567–583, Jan. 2008.
[24] C. S. Ouyang and S. J. Lee, "An improved learning algorithm for rule refinement in neuro-fuzzy modeling," in *Proc. 3rd Int. Conf. Knowl.-based Intell. Inf. Eng. Syst.*, 1999, pp. 238–241.
[25] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, Dec. 1999.
[26] J. M. Mendel and R. I. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 117–127, Apr. 2002.
[27] C. Wagner and H. Hagras, "Toward general type-2 fuzzy logic systems based on zSlices," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 637–660, Aug. 2010.
[28] D. Wu and J. M. Mendel, "Computing with words for hierarchical decision making applied to evaluating a weapon system," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 441–460, Jun. 2010.
[29] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic System: Introduction and New Directions*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.
[30] X. X. Zhang, H. X. Li, and C. K. Qi, "Spatially constrained fuzzy-clustering-based sensor placement for spatiotemporal fuzzy-control system," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 5, pp. 946–957, Oct. 2010.
[31] J. M. Mendel, "Comments on 'A-plane representation for type-2 fuzzy sets: Theory and applications'," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 1, pp. 229–230, Feb. 2010.
[32] A. Niewiadomski, "On finity, countability, cardinalities, and cylindric extensions of type-2 fuzzy sets in linguistic summarization of databases," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 3, pp. 532–545, Jun. 2010.
[33] J. Aisbett, J. T. Rickard, and D. G. Morgenthaler, "Type-2 fuzzy sets as functions on spaces," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 841–844, Aug. 2010.
[34] H. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, Aug. 2004.
[35] E. Kayacan, O. Cigdem, and O. Kaynak, "Sliding mode control approach for online learning as applied to type-2 fuzzy neural networks and its experimental evaluation," *IEEE Trans. Ind. Electron.*, vol. 59, no. 9, pp. 3510–3520, Sep. 2012.
[36] C. F. Juang and C. H. Hsu, "Reinforcement ant optimized fuzzy controller for mobile-robot wall-following control," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3931–3940, Oct. 2009.
[37] C. S. Lee, M. H. Wang, and H. Hagras, "A type-2 fuzzy ontology and its application to personal diabetic-diet recommendation," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 2, pp. 374–395, Apr. 2010.
[38] M. A. Khanesar, E. Kayacan, M. Teshnehlab, and O. Kaynak, "Analysis of the noise reduction property of type-2 fuzzy logic system using a novel type-2 membership function," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1395–1406, Oct. 2011.
[39] C. H. Wang, C. S. Cheng, and T. T. Lee, "Dynamical optimal training for interval type-2 fuzzy neural network," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1462–1477, Jun. 2004.
[40] Y. C. Lin and C. H. Lee, "Type-2 fuzzy neural network systems and learning," *Int. J. Comput. Cognit.*, vol. 1, no. 4, pp. 79–90, Dec. 2003.
[41] H. Hagras, "Comments on dynamical optimal training for interval type-2 fuzzy neural network (T2FNN)," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 5, pp. 1206–1209, Oct. 2006.
[42] J. M. Mendel, "Computing derivatives in interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, Feb. 2004.
[43] C. Y. Yeh, W. R. Jeng, and S. J. Lee, "Data-based system modeling using a type-2 fuzzy neural network with a hybrid learning algorithm," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2296–2309, Dec. 2011.
[44] C. F. Juang and Y. W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, Dec. 2008.
[45] J. Y. Chang, Y. Y. Lin, M. F. Han, and C. T. Lin, "A functional-link based interval type-2 compensatory fuzzy neural network for nonlinear system modeling," in *Proc. Int. Conf. Fuzzy Syst.*, Jun. 27–30, 2011, pp. 939–943.
[46] C. F. Juang, R. B. Huang, and Y. Y. Lin, "A recurrent self-evolving interval type-2 fuzzy neural network for dynamic system processing," *IEEE Trans. Fuzzy Syst.*, vol. 17, no. 5, pp. 1092–1105, Oct. 2009.
[47] J. R. Castro, O. Castillo, P. Melin, and A. Rodrigue-Diaz, "A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks," *Inf. Sci.*, vol. 179, no. 13, pp. 2175–2193, Jun. 2009.
[48] R. H. Abiyev and O. Kaynak, "Type 2 fuzzy neural structure for identification and control of time-varying plants," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4147–4159, Dec. 2010.
[49] F. J. Lin and P. H. Chou, "Adaptive control of two-axis motion control system using interval type-2 fuzzy neural network," *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 178–193, Jan. 2009.
[50] O. Uncu and I. B. Truksen, "Discrete interval type 2 fuzzy system models using uncertainty in learning parameters," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 90–106, Feb. 2007.
[51] M. A. Khanesar, E. Kayacan, M. Teshnehiab, and O. Kaynak, "Extended Kalman filter based learning algorithm for type-2 fuzzy logic systems and its experimental evaluation," *IEEE Trans. Ind. Electron.*, vol. 59, no. 11, pp. 4443–4455, Nov. 2012.
[52] L. Guo, "Estimating time-varying parameters by the Kalman filter based algorithm: Stability and convergence," *IEEE Trans. Autom. Control*, vol. 35, no. 2, pp. 141–147, Feb. 1990.
[53] J. J. Rubio and W. Yu, "Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm," *Neurocomputing*, vol. 70, no. 13–15, pp. 2460–2466, Aug. 2007.
[54] W. Li and Y. Hori, "An algorithm for extracting fuzzy rules based on RBF neural network," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1269–1276, Jun. 2006.
[55] S. C. Kremer, "On the computational power of Elman-style recurrent networks," *IEEE Trans. Neural Netw.*, vol. 6, no. 4, pp. 1000–1004, Jul. 1995.
[56] C. H. Lu and C. C. Tsai, "Generalized predictive control using recurrent fuzzy neural networks for industrial processes," *J. Process Control*, vol. 17, no. 1, pp. 83–92, Jan. 2007.
[57] C. T. Lin and C. F. Juang, "An adaptive neural fuzzy filter and its applications," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 27, no. 4, pp. 635–656, Aug. 1997.
[58] O. Castillo and P. Melin, "Adaptive noise cancellation using type-2 fuzzy logic and neural networks," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jul. 2004, vol. 2, pp. 1093–1098.
[59] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification," *Fuzzy Sets Syst.*, vol. 83, no. 3, pp. 325–339, Nov. 1996.
[60] J. Kim and N. K. Kasabov, "HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamic systems," *Neural Netw.*, vol. 12, no. 9, pp. 1301–1319, Nov. 1999.
[61] D. Nauk and R. Kruse, "Neuro-fuzzy systems for function approximation," *Fuzzy Sets Syst.*, vol. 101, no. 2, pp. 261–271, Jan. 1999.
[62] S. Wu and M. J. Er, "Dynamic fuzzy neural networks—A novel approach to function approximation," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 30, no. 2, pp. 358–364, Apr. 2000.
[63] Y. Chen, B. Yang, and J. Dong, "Time-series prediction using a local linear wavelet neural network," *Neurocomputing*, vol. 69, no. 4–6, pp. 449–465, Jan. 2006.
[64] Y. Takahashi, "Adaptive predictive control of nonlinear time varying systems using neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 1464–1468.
[65] Y. C. Lin and C. H. Lee, "System identification and adaptive filter using a novel fuzzy neuro system," *Int. J. Comput. Cognit.*, vol. 5, no. 1, pp. 15–26, Mar. 2007.

Yang-Yin Lin received the B.S. degree from the Department of Electronics Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan, in 2005 and the M.S. degree from the Institute of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan, in 2008. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, National Chiao Tung University, Hsinchu, Taiwan.

His current research interests include computational intelligence, type-2 fuzzy neural networks, and field-programmable gate array chip design.

Jyh-Yeong Chang (S'84–M'86) received the B.S. degree in control engineering and the M.S. degree in electronic engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1976 and 1980, respectively, and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, NC, USA, in 1987.

During 1976–1978 and 1980–1982, he was a Research Fellow at the Chung Shan Institute of Science and Technology, Lung-Tan, Taiwan. Since 1987, he has been an Associate Professor with the Department of Electrical Engineering, National Chiao Tung University, where he is currently a Professor. His current research interests include neural fuzzy systems, video processing and surveillance, and e-health systems.

Chin-Teng Lin (S'88–M'91–SM'99–F'05) received the B.S. degree from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1986 and the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1989 and 1992, respectively.

He is currently the Provost, Chair Professor of Electrical and Computer Engineering, and Director of the Brain Research Center, NCTU. He has published more than 120 journal papers in the areas of neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuro-engineering, including approximately 74 IEEE journal papers. He is the coauthor of *Neural Fuzzy Systems* (Prentice-Hall, 1996) and the author of *Neural Fuzzy Control Systems with Structure and Parameter Learning* (World Scientific, 1994).

Dr. Lin serves as the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He also served on the Board of Governors of the IEEE Circuits and Systems Society in 2005–2008, IEEE Systems, Man, and Cybernetics Society in 2003–2005, and IEEE Computational Intelligence Society in 2008–2010 and was the Chair of the IEEE Taipei Section in 2009–2010. He served as the Deputy Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II in 2006–2008.